Make Sure Critical Software Performs in its Intended Environment

The Problem:

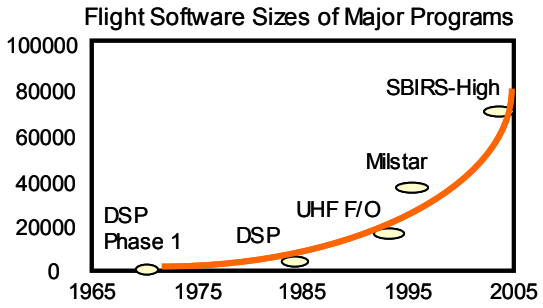
The 1996 maiden flight of a launch vehicle ended in a crash because a software error disabled the inertial reference system.

The Cause:

The launcher's flight control system, which had derived considerable heritage from the previous generation, used two identical inertial reference controllers, including a "hot" stand-by.

One function inherited from the legacy software computed the platform alignment before launch. This function was no longer needed in the new generation.

The new rocket flew a different trajectory, creating an alignment bias that was too large for the legacy code to compute. An "operand error exception" occurred as a result.



As software takes over many functions that used to be controlled by hardware, code sizes increase almost exponentially. Software reliability thus poses a growing challenge and warrants more quality assurance efforts.

Such errors are common, and are typically handled by software (for example, by inserting "likely" values). Unfortunately, although the programmers did identify the alignment bias input as one of the several variables capable of causing operand errors, they chose to leave it unprotected, probably supposing that there would be large safety margins.

More tragically, the system was designed in the belief that any fault would be due to random hardware problems, and should be handled by an equipment swap. Thus, when the software detected the errant and irrelevant exception, it halted the active controller and switched to the backup. Of course, the backup immediately encountered the same error exception, and also shut down. The launch vehicle in essence destroyed itself even though both controllers worked perfectly.

Lessons Learned:

- Hardware redundancy does not necessarily protect against software faults.
- Mission-critical software failures should be included in system reliability and fault analysis.
- Software specifications should always include specific operational scenarios.
- Software reuse should be thoroughly analyzed to ensure suitability in a new environment, and all associated documentation, especially assumptions, should be reexamined.
- Extensive testing should be performed at every level, from unit through system test, using realistic operational and exception scenarios.

For more technical information, call Suellen Eslinger at (310) 336-2906.

For comments on the Aerospace Lessons Learned Program, including background specifics, call Paul Cheng at (310) 336-8222.